Distributed interrupts mechanism implementation and investigation by modeling on SDL and SystemC

Fourth FRUCT seminar

Alexey Rabin,
Irina Lavrovskaya,
Ludmila Onischenko
Artur Eganyan

SUAI, St. Petersburg, Russia

29-31 October 2008

Outline

- Distributed interrupts overview
- Distribution recovery in case of errors
- Specification and description language (SDL), definition & overview
- SDL model of SpaceWire distributed interrupts
- SDL abilities
- Investigation by SpaceWire Network Functional SystemC model
- SystemC abilities
- Joint use of SDL and SystemC

Distributed Interrupts overview

- In terminal nodes of a distributed computing system can occur some events about which it is necessary **to report** to other terminal nodes quickly and **to receive a confirmation** about these events' processing
- By transmission of information about such event in form of ordinary message several difficulties can occur
- For overcoming of them in SpaceWire standard a special opportunity for transmission of such events is provided
- These are **distributed interrupts (Interrupt codes)** and their **confirmations – Interrupt_Acknowledge codes**
- Interrupt code and Interrupt_Acknowledge code are special control characters, which have higher priority than data

Distributed Interrupts overview

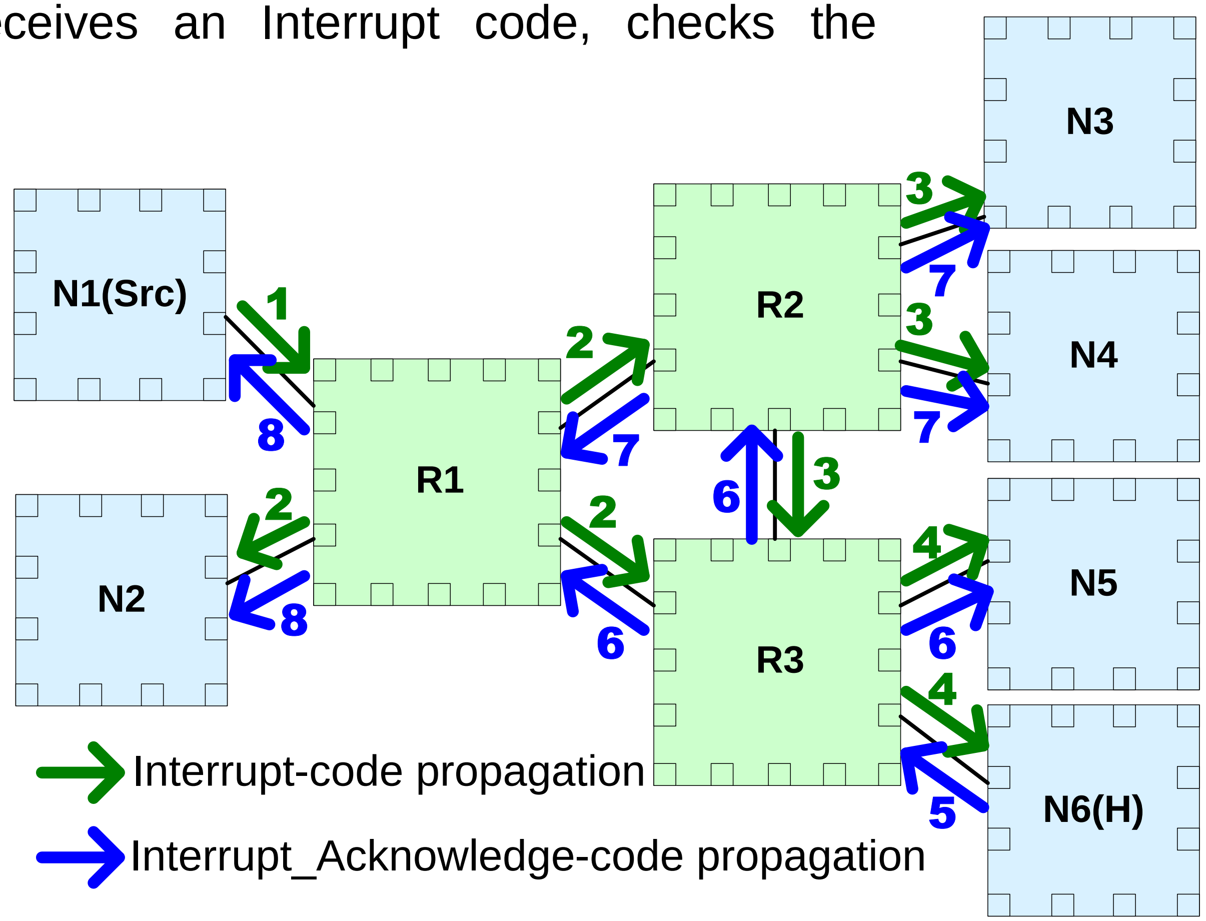
Nodes are Interrupt code sources and handlers (Ex: N1 is source and N6 is handler)

Node's link controllers and routers contain 32-bit Interrupt Source Register (ISR)

Router: a link interface receives an Interrupt code, checks the corresponding bit in the ISR:

If the bit is '0' it sets the ISR bit to '1' and the signal propagates to all the router output ports (except the port that has issued the signal).

If the corresponding bit in the ISR is equal to '1' the Interrupt code will be ignored


N1(Src)
N2
R1
R2
R3
N3
N4
N5
N6(H)
1
2
2
2
3
3
3
4
4
5
6
6
6
7
7
7
8
8
Interrupt-code propagation
Interrupt_Acknowledge-code propagation

Node:

A subsequent Interrupt code with the same interrupt source identifier can be sent by the link only after receipt of an corresponding Interrupt_Acknowledge.

Distribution recovery in case of errors

- To ensure **tolerance against faults** each ISR in a node and in a router has a timer per ISR’s bit.
- A timer starts at the receipt of an Interrupt code with corresponding five-bit interrupt source identifier and **resets at receipt of an Interrupt_Acknowledge** code with the same interrupt source identifier.
- In case of **timeout before the Interrupt_Acknowledge** receiption, the ISR timeout event arises; the corresponding ISR bit should be reset to ‘0’.

Specification and Description Language (SDL)

- **Definition**

  - Specification and description language (SDL) is an object-oriented, formal language defined by The International Telecommunications Union-Telecommunications Standardization Sector (ITU–T) as recommendation Z.100;

  - The language is intended for the specification of complex, event-driven, real-time and interactive applications involving many concurrent activities that communicate using discrete signals.

- **Overview**

  - Now it is increasingly accepted within a steadily growing range of industrial segments that the best way to meet the needs of these systems is through formal methods;

  - Formal methods should be internationally standardized;

  - Telecommunications software engineers have developed such methods and tools for the development of complex real-time software;

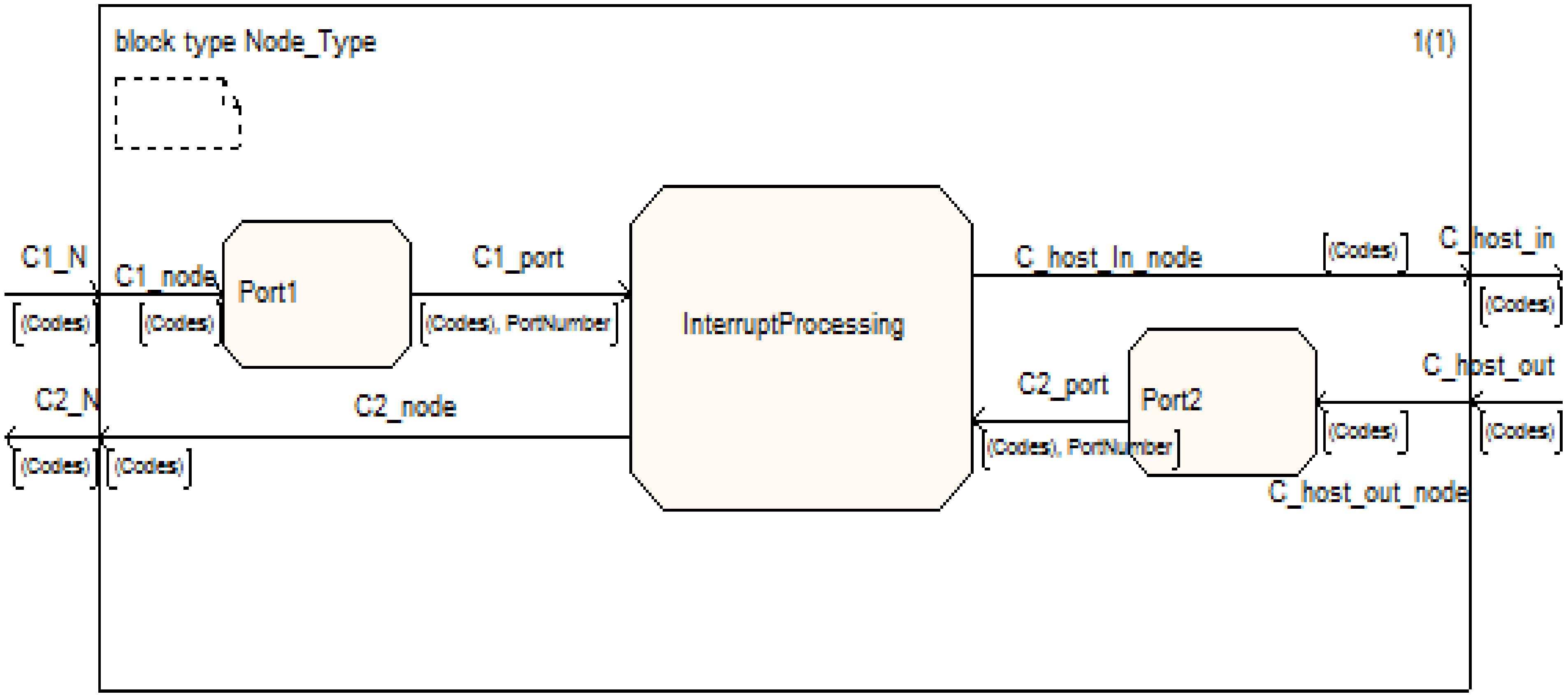
  - The benefit of SDL is its ability to describe the structure, behavior and data of a system.

SDL model of SpaceWire distributed interrupts

- Includes description of general elements of the SpaceWire network: a node, a router, a link
- Allows to create networks of any construction and difficulty

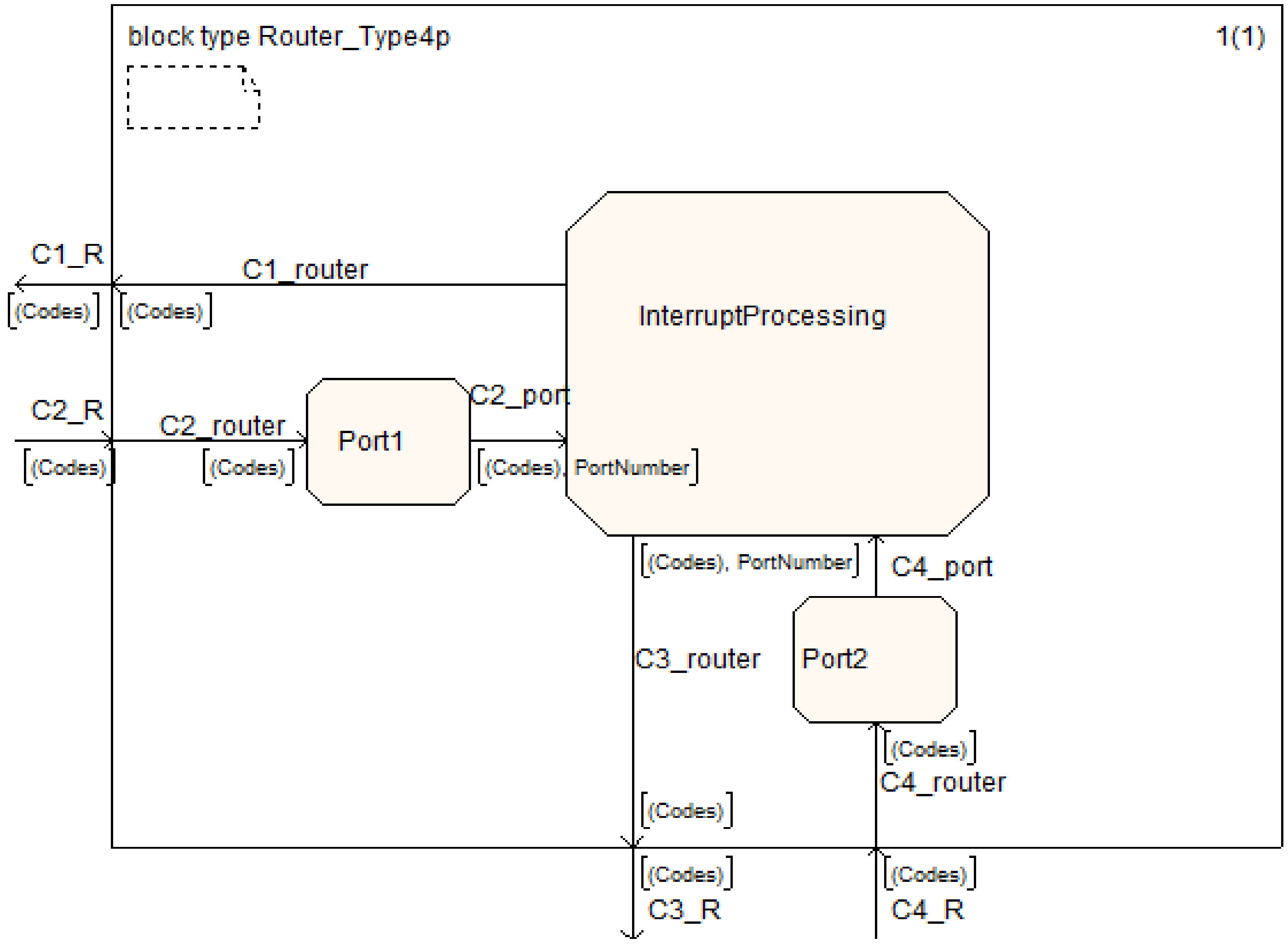
Node

General description of the block Node functionality: 3 processes implement two ports and the main process InterruptProcessing, which processes the interrupts



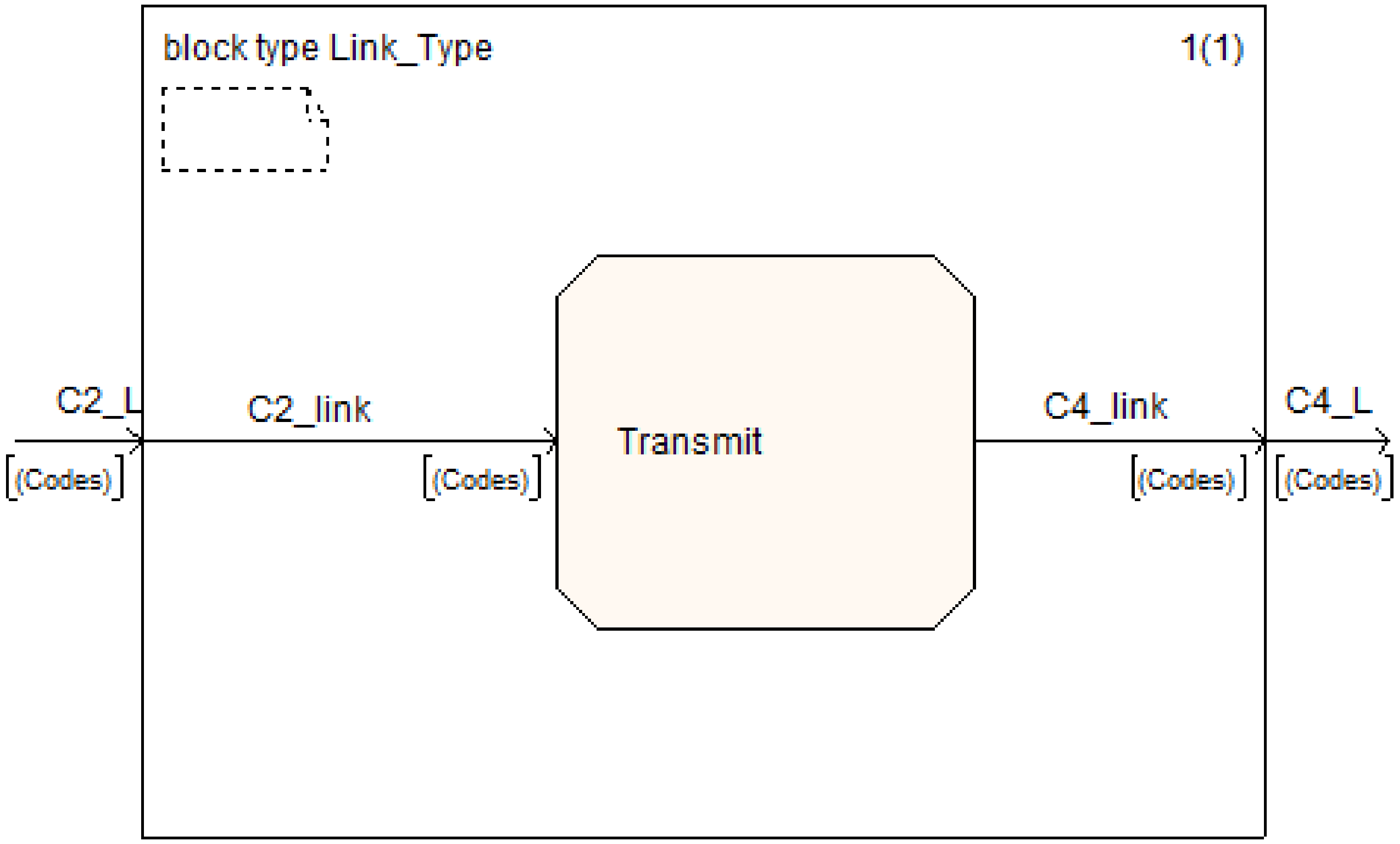
Router

General description of the block Router functionality: 3 processes implement two ports and the main process InterruptProcessing,which processes
the interrupts



Link

General description of the block Link functionality: process Transit, which receives signals and sends them after some period of time



SDL abilities

- SDL ToolSuite gives an ability **to implement** specifications. A distributed interrupts system implementation on the SDL allows to have a reference implementation for it and **check** how Interrupt codes and Interrupt_Acknowledge codes are sent through the network.
- It is possible **to investigate** distributed interrupts mechanism, check the correctness of the realization by **verification**.
- Opportunity **to model** some difficult in investigation situations with lost of data, errors in links, distribution recovery in case of errors etc.

Investigation by SpaceWire Network Functional model

The SpaceWire Network Functional model (SpWNM)

- includes a description of basic SpaceWire network elements like node, routing switch and link,
- allows to assemble a SpaceWire interconnection system of required structure,
- implements wormhole routing, generation and transmission of data packets, time flow and distributed interrupts mechanisms.

The SystemC model consists of

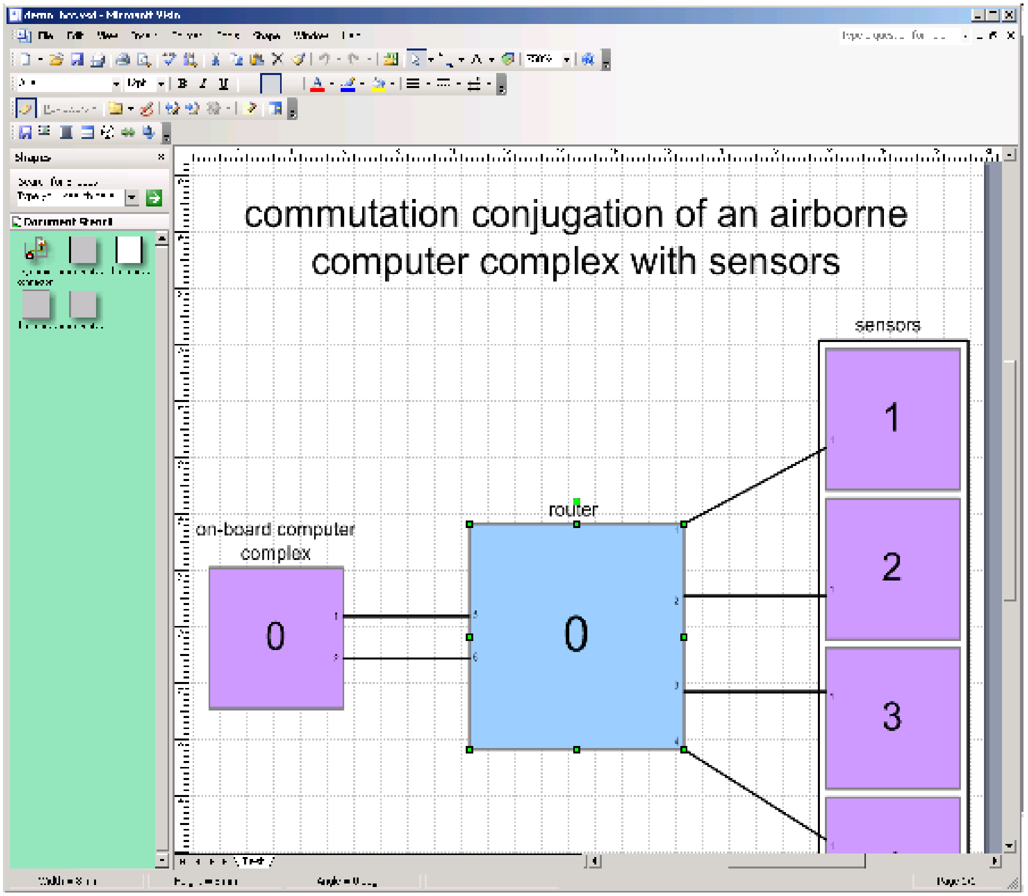
- libraries for systems building in MS Visio
- modeling core for built systems
- parser and analyzer for modeling results

In the SpWNM any researched system is represented as a set of basic SpaceWire network devices – terminal nodes and routers. These devices are linked with each other by bidirectional communication channels.

Libraries for systems building in MS Visio

System designing in MS Visio with use of the libraries given in a complex:

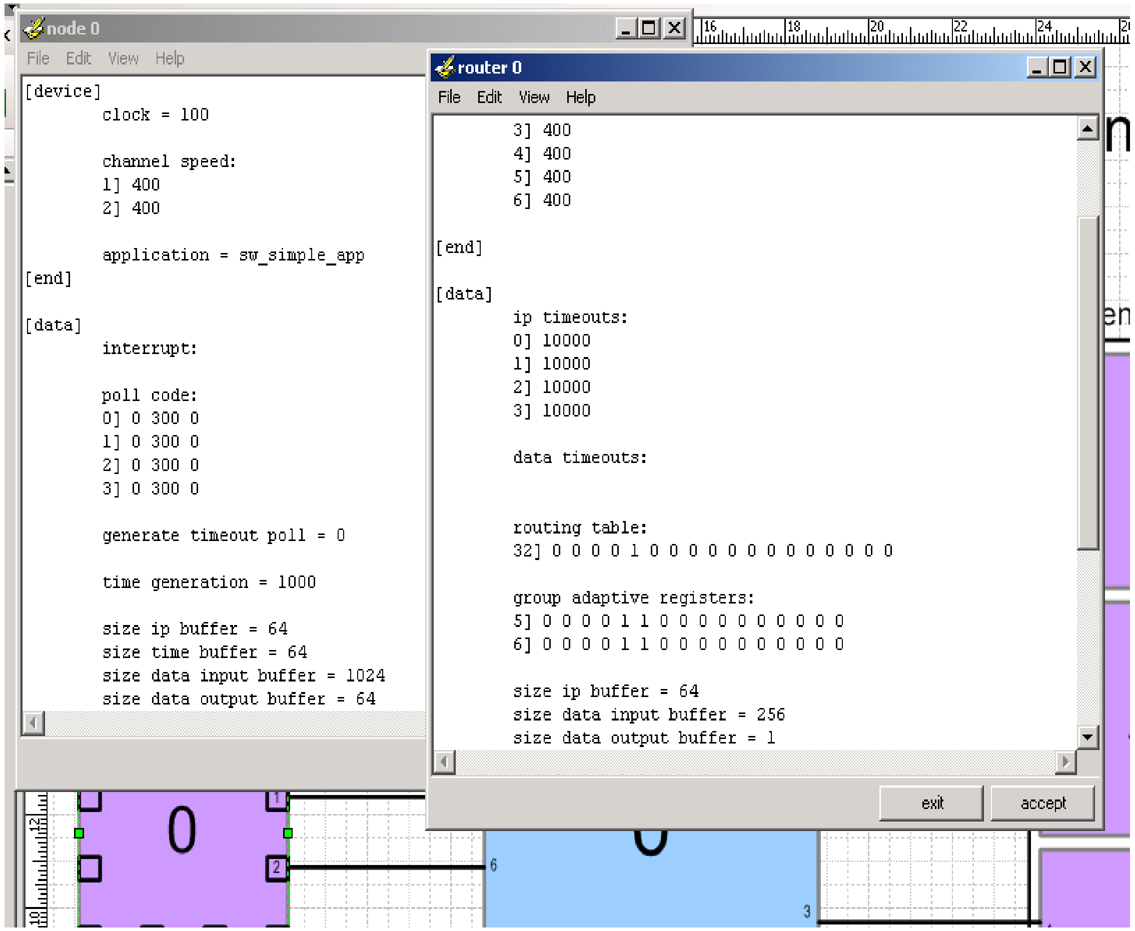
- User chooses devices of different types,
- establishes communications between them
- and sets parametres of each device


commutation conjugation of an airborne computer complex with sensors
sensors
1
2
3
router
0
on-board computer complex
0

Libraries for systems building in MS Visio

System designing in MS Visio with use of the libraries given in a complex:

- User chooses devices of different types,
- establishes communications between them
- and sets parametres of each device

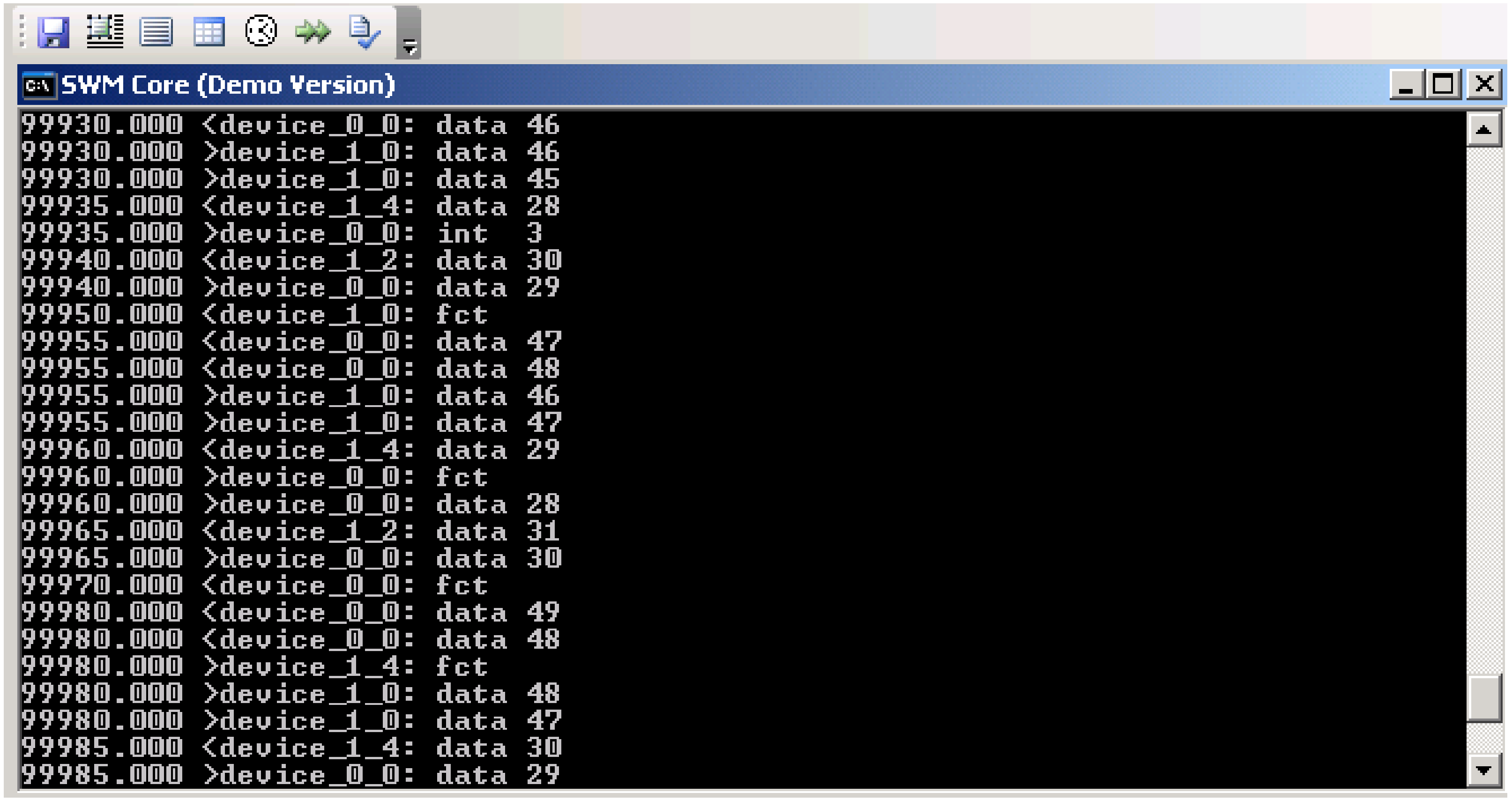


```
[device]
        clock = 100

        channel speed:
        1] 400
        2] 400

        application = sw_simple_app
[end]

[data]
        interrupt:

        poll code:
        0] 0 300 0
        1] 0 300 0
        2] 0 300 0
        3] 0 300 0

        generate timeout poll = 0

        time generation = 1000

        size ip buffer = 64
        size time buffer = 64
        size data input buffer = 1024
        size data output buffer = 64
```

```
        3] 400
        4] 400
        5] 400
        6] 400

[end]

[data]
        ip timeouts:
        0] 10000
        1] 10000
        2] 10000
        3] 10000

        data timeouts:


        routing table:
        32] 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0

        group adaptive registers:
        5] 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0
        6] 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0

        size ip buffer = 64
        size data input buffer = 256
        size data output buffer = 1
```

Modeling core for built systems

Booting of the built system by a modeling core with the installation parameters. After that the system starts the modeling for a specified time.



```
99930.000 <device_0_0: data 46
99930.000 >device_1_0: data 46
99930.000 >device_1_0: data 45
99935.000 <device_1_4: data 28
99935.000 >device_0_0: int  3
99940.000 <device_1_2: data 30
99940.000 >device_0_0: data 29
99950.000 <device_1_0: fct
99955.000 <device_0_0: data 47
99955.000 <device_0_0: data 48
99955.000 >device_1_0: data 46
99955.000 >device_1_0: data 47
99960.000 <device_1_4: data 29
99960.000 >device_0_0: fct
99960.000 >device_0_0: data 28
99965.000 <device_1_2: data 31
99965.000 >device_0_0: data 30
99970.000 <device_0_0: fct
99980.000 <device_0_0: data 49
99980.000 <device_0_0: data 48
99980.000 >device_1_4: fct
99980.000 >device_1_0: data 48
99980.000 >device_1_0: data 47
99985.000 <device_1_4: data 30
99985.000 >device_0_0: data 29
```

SystemC abilities

- Long time modeling of the system work
- It is easy to get some average values, to make relations of output characteristics from input values, to choose parameters for certain network structure
- Flexible system of log generation allows to make compact notes and only necessary data

Joint use of SDL and SystemC

allows

- Get some data
- Compare results
- Get statistics
- Investigate specifications

Joint use of SDL and SystemC

- To use the models not in parallel, but one model will work on the results of the second model (for example, SDL model will work with parameters of the SystemC model).
- Parallel modeling. Two realizations will themselves exchange the information

Models are suitable for

SDL

- Modeling of spontaneous changes of the ISR register
- Errors in links
- Built in visualization means

SystemC

- Modeling of networks with large number of elements
- Modeling of long time working of system
- Flexible system of log generation

Thank You